APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/662,258	09/14/2000	Judith E. Schwabe	SUN-P4175	1082

7590 12/04/2003		EXAMINER
THELEN REID & PRIEST LLP		GODDARD, BRIAN D
ATTN: David B. Ritchie		

ART UNIT	PAPER NUMBER
2171	

DATE MAILED: 12/04/2003

THELEN REID & PRIEST LLP
ATTN: David B. Ritchie
P O Box 640640
San Jose, CA 95164-0640

Please find below and/or attached an Office communication concerning this application or proceeding.

Application No.

09/662,258

Applicant(s)

SCHWABE, JUDITH E.

Examiner

Brian Goddard

Art Unit

2171

Office Action Summary

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) FROM
THE MAILING DATE OF THIS COMMUNICATION.
- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed
  after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
- Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any
  earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *22 September 2003*.

2a)☒ This action is **FINAL**.   2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is
   closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-21* is/are pending in the application.

   4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-21* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *22 September 2003* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

   Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

   Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. §§ 119 and 120

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

   a)☐ All b)☐ Some * c)☐ None of:

   1.☐ Certified copies of the priority documents have been received.

   2.☐ Certified copies of the priority documents have been received in Application No. _____.

   3.☐ Copies of the certified copies of the priority documents have been received in this National Stage
      application from the International Bureau (PCT Rule 17.2(a)).

   * See the attached detailed Office action for a list of the certified copies not received.

13)☒ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. § 119(e) (to a provisional application)
   since a specific reference was included in the first sentence of the specification or in an Application Data Sheet.
   37 CFR 1.78.

   a) ☐ The translation of the foreign language provisional application has been received.

14)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. §§ 120 and/or 121 since a specific
   reference was included in the first sentence of the specification or in an Application Data Sheet. 37 CFR 1.78.

Attachment(s)

1)☐ Notice of References Cited (PTO-892)
2)☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3)☐ Information Disclosure Statement(s) (PTO-1449) Paper No(s) _____.

4)☐ Interview Summary (PTO-413) Paper No(s). _____.
5)☐ Notice of Informal Patent Application (PTO-152)
6)☐ Other: _____.

DETAILED ACTION

1.   This communication is responsive to Amendment B, filed 22 September 2003.

2.   Claims 1-21 are pending in this application. Claims 1, 3, 5, 8, 10, 12, 15, 17 and

19 are independent claims. In Amendment B, claims 1-6, 8-13 and 15-20 were

amended. This action is made Final.

Claim Rejections - 35 USC § 103

The text of those sections of Title 35, U.S. Code not included in this action can

be found in a prior Office action.

3.   Claims 1-6, 8-13 and 15-20 are rejected under 35 U.S.C. 103(a) as being

unpatentable over U.S. Patent No. 5,408,665 to Fitzgerald in view of U.S. Patent No.

6,526,571 to Aizikowitz et al.

Referring to claim 1, Fitzgerald teaches a system and method for listing public

elements in a library as claimed. See Figures 3-4 and the corresponding portions of

Fitzgerald's specification for this disclosure. Refer also to claims 1 and 6 for more

details of this disclosure. In particular, Fitzgerald teaches a method for representing an

application programming interface (API) definition for a programming language library

[260], said method comprising:

creating [Librarian 265 creates] a list [Standard Dictionary 360 (also 430): 'a list

of the library's public symbols and module names' (Column 8, lines 51-59)] of public

elements [library object modules (See Fig. 3B)] in said programming language library,

each of said public elements including a sublist [Dependency List 445] of all public

hierarchically-related elements that are a parent of the element ['each module it needs'

(Column 11, lines 17-25) See also Column 3, lines 13-25]; and

storing [stored in Library File 410 (See Fig. 4A)] said list.

Fitzgerald does not explicitly state that the library (260) is an object-oriented

library as claimed, and thus does not expressly state that the elements are

hierarchically related. However, Fitzgerald does state that in the preferred embodiment,

the programming language specific to the system is Borland C++. See column 5, lines

46-59 for this disclosure. C++ being an object-oriented language, this provides direct

suggestion for using an object-oriented library for Fitzgerald's library as claimed.

Furthermore, one can infer that Fitzgerald's library is object-oriented because it stores

objects. See Figures 3B-4A and the corresponding portions of Fitzgerald's specification

for this disclosure.

Aizikowitz teaches a system and method similar to that of Fitzgerald, wherein a

class dependency hierarchy is generated from an object oriented library. See Figures 1

& 2 and the corresponding portions of Aizikowitz' specification for this disclosure. In

particular, Aizikowitz teaches the practice of creating a class hierarchy (CHG) for

classes and interfaces of a Java package (object-oriented library).

It would have been obvious to one of ordinary skill in the art at the time the

invention was made to apply Fitzgerald's method of creating a list of public elements

reflecting their dependencies to the object-oriented library of Aizikowitz in order to

derive the object-oriented library's dependency hierarchy in a list structure as claimed.

One would have been motivated to do so because of the suggestions provided by Fitzgerald as above.

Referring to claim 2, the system and method of Fitzgerald in view of Aizikowitz as applied to claim 1 above discloses the invention as claimed. Aizikowitz' object-oriented library (as applied to Fitzgerald) is a Java package as claimed. See Figure 1 and column 2, line 50 et seq. of Aizikowitz' specification for this disclosure. Furthermore, Aizikowitz' public elements (as applied to Fitzgerald) comprise classes and interfaces as claimed. See Figure 2; column 2, lines 58-59; and column 3, lines 40-46 of Aizikowitz' specification for this disclosure. Finally, Aizikowitz' public hierarchically-related elements (as applied to Fitzgerald) comprise public superclasses and public superinterfaces of said classes and said interfaces as claimed. See Figure 2 and column 7, lines 25-30 of Aizikowitz' specification, in light Fitzgerald's disclosure of the Dependency List in the combination above, for this disclosure.

Claims 3 and 4 are rejected on the same basis as claim 2 above. See the discussions regarding claims 1 and 2 above for the details of the disclosure of the claimed method and the detailed limitations as well.

Referring to claim 5, the system and method of Fitzgerald in view of Aizikowitz as applied to claim 1 above discloses the invention as claimed. See Figure 6 and the corresponding portion of Fitzgerald's specification for this disclosure. In particular, Fitzgerald (as modified by Aizikowitz) teaches a method for determining a program hierarchy, said method comprising:

receiving [Step 601] an application programming interface (API) definition file

[Standard and Extended Dictionaries] for an object-oriented library, said API definition

file including...[See the discussion regarding claim 1 above]; and

traversing the program hierarchy through the dependency list [See Fig. 6C].

Fitzgerald (as modified by Aizikowitz) does not explicitly teach the step of

"indicating a first public element is a direct parent of a second public element" as

claimed. However, looking at the structure of Fitzgerald's (as modified by Aizikowitz)

Extended Dictionary described above with regard to claims 1 and 2, one can infer that

the direct parent of a specific module (public element) is represented in the sublist

(dependency list) of that module, but is not represented in the sublist of any other

modules listed in that module's sublist. In other words, in order to traverse Fitzgerald's

(as modified by Aizikowitz) hierarchy, a first module's direct parent can be found by

searching that first module's sublist to find the second module that is not listed in the

sublist for any other module in the first module's sublist.

It would have been obvious to one of ordinary skill in the art at the time the

invention was made to program Fitzgerald's (as modified by Aizikowitz) system to

traverse the Extended Dictionary's hierarchy to find a first module's direct parent by

searching that first module's sublist to find the second module that is not listed in the

sublist for any other module in the first module's sublist as claimed. One would have

been motivated to do so because this method is easily inferred from the structure of the

Extended Dictionary, and seems to be the only method for traversing the hierarchy

possible.

Claim 6 is rejected on the same basis as claim 2 above, in light of the basis for

claim 5. See the discussions regarding claims 1, 2 and 5 above for the details of this

disclosure.

Claims 8-13 are rejected on the same basis as claims 1-6 respectively. See the

discussions regarding claims 1-6 above for the details of this disclosure.

Claims 15-20 are rejected on the same basis as claims 1-6 respectively. See the

discussions regarding claims 1-6 above for the details of this disclosure.

4.   Claims 7, 14 and 21 are rejected under 35 U.S.C. 103(a) as being unpatentable

over Fitzgerald in view of Aizikowitz as applied to claims 5, 12 and 19 above, and

further in view of U.S. Patent No. 5,974,255 to Gossain et al.

Referring to claim 7, the system and method of Fitzgerald in view of Aizikowitz as

applied to claim 5 above does not explicitly disclose the steps of comparing two

reconstructed program hierarchies and indicating an error when they are inconsistent as

claimed. However, Aizikowitz does disclose the need to maintain integrity of the

program hierarchy in order to maintain the signed and sealed status of the package.

See the Background and Summary of the Invention sections of Aizikowitz' specification

for this disclosure. This provides suggestion for examining the hierarchy of an API with

an expected hierarchy to maintain consistency for the signed and sealed status.

Gossain discloses a method for testing the inheritance hierarchy of an object-

oriented class structure by comparing the active hierarchy to a test hierarchy stored

within the system. See the Figure and the Detailed Description of the Drawing section

for this disclosure. Refer specifically to column 3, lines 6-14. Gossain teaches the two

claimed steps as follows:

Comparing [step 18] a first program hierarchy [hierarchy of class under test (11)]

with a second program hierarchy [test class hierarchy (12)]; and

Indicating an error [Column 3, lines 9-10] when said first program hierarchy is

inconsistent ['when a difference between the current state and expected state...is

detected' (Column 3, lines 7-8)] with said second program hierarchy.

It would have been obvious to one of ordinary skill in the art at the time the

invention was made to incorporate Gossain's method for testing class hierarchies into

Fitzgerald's (as modified by Aizikowitz) system such that the system would compare the

hierarchy reconstructed from an Extended Dictionary for one library with the hierarchy

reconstructed from a test Extended Dictionary, and indicate an error when the two

hierarchies were inconsistent. One would have been motivated to do so because of

Aizikowitz' suggestion described above.

Claims 14 and 21 are each rejected on the same basis as claim 7 above. See

the discussion regarding claim 7 for the details of this disclosure.

5.   Claims 1-4, 8-11 and 15-18 are rejected under 35 U.S.C. 103(a) as being

unpatentable over U.S. Patent No. 6,230,314 to Sweeney et al. in view of Aizikowitz.

Referring to claim 1, Sweeney discloses a system and method for generating an

object-oriented program inheritance listing. See Figures 3, 4 & 7 and the corresponding

portions of Sweeney's specification for this disclosure. In particular, Sweeney teaches a

method for representing an application programming interface (API) definition for an

object-oriented program, said method comprising:

creating [Steps 703-707] a list [Class Hierarchy (See Fig. 3 and column 3, line 59

– column 4, line 4)] of public elements [set of classes] in said object-oriented program,

each of said public elements ['for every class' (column 4, line 2)] including a sublist of all

public hierarchically-related elements that are a parent of the element ['the set of base

classes it inherits from is specified' (column 4, lines 2-3)]; and

storing said list [See column 19, lines 56-62].

Sweeney does not explicitly disclose that the object-oriented program used for

generating the API definition is an object-oriented library as claimed. However,

Sweeney does disclose the use and importance of object-oriented libraries in the

background of the invention section (See column 1, lines 11-24). This provides

suggestion for applying Sweeney's method to an object-oriented library.

Aizikowitz teaches a system and method similar to that of Sweeney, wherein a

class dependency hierarchy is generated from an object oriented library. See Figures 1

& 2 and the corresponding portions of Aizikowitz' specification for this disclosure. In

particular, Aizikowitz teaches the practice of creating a class hierarchy (CHG) for

classes and interfaces of a Java package (object-oriented library).

It would have been obvious to one of ordinary skill in the art at the time the

invention was made to apply Sweeney's method of creating a list of public elements

reflecting their dependencies to the object-oriented library of Aizikowitz in order to

derive the object-oriented library's dependency hierarchy in a list structure as claimed.

One would have been motivated to do so because of the suggestions provided by Sweeney as above.

Referring to claim 2, the system and method of Sweeney in view of Aizikowitz as applied to claim 1 above discloses the invention as claimed. Aizikowitz' object-oriented library (as applied to Sweeney) is a Java package as claimed. See Figure 1 and column 2, line 50 et seq. of Aizikowitz' specification for this disclosure. Furthermore, Aizikowitz' public elements (as applied to Sweeney) comprise classes and interfaces as claimed. See Figure 2; column 2, lines 58-59; and column 3, lines 40-46 of Aizikowitz' specification for this disclosure. Finally, Aizikowitz' public hierarchically-related elements (as applied to Sweeney) comprise public superclasses and public superinterfaces of said classes and said interfaces as claimed. See Figure 2 and column 7, lines 25-30 of Aizikowitz' specification, in light Sweeney's disclosure of the Dependency List in the combination above, for this disclosure.

Claims 3 and 4 are rejected on the same basis as claim 2 above. See the discussions regarding claims 1 and 2 above for the details of the disclosure of the claimed method and the detailed limitations as well.

Claims 8-11 are rejected on the same basis as claims 1-4 respectively. See the discussions regarding claims 1-4 above for the details of this disclosure.

Claims 15-18 are rejected on the same basis as claims 1-4 respectively. See the discussions regarding claims 1-4 above for the details of this disclosure.

Response to Arguments

6.   Applicant's arguments filed 22 September 2003 have been fully considered but they are not persuasive.

Referring to applicant's remarks on pages 13-17 regarding the first 35 U.S.C § 103 rejection: Applicant argued that Fitzgerald does not disclose object-oriented libraries, and that neither reference (although only discussing Fitzgerald) discloses creating a list of public elements...where each of the public elements include a sublist of all public *hierarchically-related* elements *that are a parent* of the element.

In response to applicant's arguments against the references individually, one cannot show nonobviousness by attacking references individually where the rejections are based on combinations of references. See *In re Keller*, 642 F.2d 413, 208 USPQ 871 (CCPA 1981); *In re Merck & Co.,* 800 F.2d 1091, 231 USPQ 375 (Fed. Cir. 1986).

Contrary to applicant's statement, the examiner has never held that Fitzgerald teaches object-oriented libraries. In fact, the opposite is true. The examiner has admitted in the previous Office action that Fitzgerald does not explicitly teach an object-oriented library. However, the examiner maintains that Fitzgerald does suggest the use of an object-oriented library through the disclosure of a C++ library, C++ being an object-oriented language. That is, Fitzgerald's use of a C++ library would have suggested the use of an object-oriented library to one of ordinary skill in the art at the time of applicant's invention because C++ was a well-known object-oriented programming language at that time. Aizikowitz does teach the use of an object-oriented library as shown above. Thus, in the combination repeated above, Fitzgerald's method

was applied to the object-oriented library of Aizikowitz because of Fitzgerald's

suggestion in order to obtain the invention as claimed. Finally, the combination (both

Fitzgerald and Aizikowitz) does teach the sublist of all public hierarchically-related

elements that are a parent of the element as shown above. In fact, the applicant admits

that, "Combining both would result in a sublist that includes public elements related to a

particular element via…a parent relationship" (Page 23 of response), thus contradicting

the argument posed here.

Referring to applicant's remarks on pages 18-19 regarding the first 35 U.S.C §

103 rejection: Applicant argued that there is no suggestion or motivation to combine the

references.

The examiner disagrees for the reasons set forth above. This argument has

been addressed in conjunction with applicant's previous argument.

Referring to applicant's remarks on pages 19-20 regarding the first 35 U.S.C §

103 rejection: Applicant argued that neither Fitzgerald nor Aizikowitz teaches the

problem or its source.

In response to applicant's argument that Fitzgerald and Aizikowitz are

nonanalogous art, it has been held that a prior art reference must either be in the field of

applicant's endeavor or, if not, then be reasonably pertinent to the particular problem

with which the applicant was concerned, in order to be relied upon as a basis for

rejection of the claimed invention. See *In re Oetiker*, 977 F.2d 1443, 24 USPQ2d 1443

(Fed. Cir. 1992). In this case, both references are considered to be in the field of

applicant's endeavor in as much as they both deal with relationships between objects in a programming environment.

Referring to applicant's remarks on pages 21-23 regarding the first 35 U.S.C § 103 rejection of claim 5: Applicant argued that the examiner impermissibly engaged in hindsight construction, and that no inference could be made regarding whether a first public element is a direct parent of a second public element by examining sublists as claimed.

In response to applicant's argument that the examiner's conclusion of obviousness is based upon improper hindsight reasoning, it must be recognized that any judgment on obviousness is in a sense necessarily a reconstruction based upon hindsight reasoning. But so long as it takes into account only knowledge which was within the level of ordinary skill at the time the claimed invention was made, and does not include knowledge gleaned only from the applicant's disclosure, such a reconstruction is proper. See *In re McLaughlin*, 443 F.2d 1392, 170 USPQ 209 (CCPA 1971).

Furthermore, applicant's statement on page 23 that combining both (Fitzgerald and Aizikowitz) would result in a sublist that includes public elements related to a particular element via a child relationship is incorrect. Specifically, in any hierarchy, the child (or children) of an element is not "needed by an element" as required by Fitzgerald's construction method. An element (or node) in a hierarchy needs only itself and its parent(s), if any, to exist. This is in accordance with the definition of a hierarchy. Therefore, in applying Fitzgerald's construction method to the hierarchy of Aizikowitz,

the sublist would not include public elements related to a particular element via a child

relationship because these child elements would not be needed by the element as

required by Fitzgerald. As such, the inference regarding whether a first public element

is a direct parent of a second public element could be made as stated above.

Referring to applicant's remarks on pages 24-27 regarding the second 35 U.S.C

§ 103 rejection: Applicant argued that none of the references disclose comparing a first

program hierarchy reconstructed from a first API definition file with a second program

hierarchy reconstructed from a second API definition file for reasons similar to those

presented in the previous argument.

The examiner disagrees for the reasons set forth above. This argument has

been addressed in conjunction with applicant's previous argument.

Referring to applicant's remarks on pages 27-30 regarding the third 35 U.S.C §

103 rejection: Applicant argued that Sweeney does not disclose creating a list of public

elements...where each of the public elements include a sublist of all public

hierarchically-related elements that are a parent of the element.

The examiner disagrees for the following reasons: Applicant bases this

argument on an example offered by Sweeney in accordance with one of the Figures.

However, Sweeney's explicit statement is that, "for ever class, the set of base classes it

inherits from is specified." (Column 4, lines 2-3) In any hierarchy, a node inherits from

all of its parents, not just its direct parents. This is also true of a class hierarchy in

object-oriented programming such as that of Sweeney. This again, is in accordance

with the definition of a hierarchy. Thus, Sweeney's "set of base classes it inherits from"

includes all public hierarchically-related elements that are a parent of the element as claimed.

Referring to applicant's remarks on page 31 regarding the third 35 U.S.C § 103 rejection: Applicant argued that there is no suggestion or motivation to combine the references.

In response to applicant's argument that there is no suggestion to combine the references, the examiner recognizes that obviousness can only be established by combining or modifying the teachings of the prior art to produce the claimed invention where there is some teaching, suggestion, or motivation to do so found either in the references themselves or in the knowledge generally available to one of ordinary skill in the art. See *In re Fine*, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988)and *In re Jones*, 958 F.2d 347, 21 USPQ2d 1941 (Fed. Cir. 1992). In this case, Sweeney's disclosure of the use and importance of object-oriented libraries in the background of the invention section (See column 1, lines 11-24) provides suggestion for applying Sweeney's method of creating a list of public elements reflecting their dependencies to the object-oriented library of Aizikowitz.

Referring to applicant's remarks on pages 32-33 regarding the third 35 U.S.C § 103 rejection: Applicant argued that neither Sweeney nor Aizikowitz teaches the problem or its source.

In response to applicant's argument that Fitzgerald and Aizikowitz are nonanalogous art, it has been held that a prior art reference must either be in the field of applicant's endeavor or, if not, then be reasonably pertinent to the particular problem

with which the applicant was concerned, in order to be relied upon as a basis for rejection of the claimed invention. See *In re Oetiker*, 977 F.2d 1443, 24 USPQ2d 1443 (Fed. Cir. 1992). In this case, both references are considered to be in the field of applicant's endeavor in as much as they both deal with relationships between objects in a programming environment.

Conclusion

7.   **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

8.   Any inquiry concerning this communication or earlier communications from the examiner should be directed to Brian Goddard whose telephone number is 703-305-7821. The examiner can normally be reached on M-F, 9 AM - 5 PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's

supervisor, Safet Metjahic can be reached on 703-308-1436. The fax phone number for

the organization where this application or proceeding is assigned is 703-872-9306.

Any inquiry of a general nature or relating to the status of this application or

proceeding should be directed to the receptionist whose telephone number is 703-305-

3900.

bdg

SAFET METJAHIC
SUPERVISORY PATENT EXAMINER
TCHNOLOGY CENTER 2100